# Partially Ordered Search Indices in the Organization of a Fixed Hierarchy

*Jorma Skyttä and Tapio Takala*

*Helsinki University of Technology*
*Department of Computer Science*
*SF-02150 Espoo, Finland*

## 1. Introduction

The mapping of even very advanced algorithms directly to hardware does not typically bring good results as these algorithms are originally designed for sequential processing. However, the power of the modern integration technology lies in its ability to produce high volumes of reasonably complex elements at moderate cost. For utilization of these possibilities the algorithms and data structures already developed must be redesigned for parallel computation.

3-D models of natural objects contain large amounts of geometric data which must be efficiently manipulated, stored and retrieved. Traditionally computational and storage oriented problems have been kept strictly apart, and hardware enhancements have been proposed for the computational problems only. This has been largely due to the general purpose computer architectures which allow acceleration of computation with additional hardware easily, but keep the I/O-facilities strictly built into the original system architecture in an unmodifiable format. This has introduced severe bottlenecks into the system capacity at I/O-level, and several attempts have been made to enhance the communication to the peripheral world with system integrated I/O-processors.

The manipulation of geometric model data involves elaborate searches through the data structure containing the model, and these operations are comparable to the computational burden introduced by the calculations of geometric transformations. A search engine architecture equivalent to geometric transformation engines can be developed for hardware acceleration of data structure operations, but the present day level of integration technology also gives a possibility to propose a unified architecture. This line of approach has a network processor as a backbone. The intention is to create local domains of the geometric model with their own processing and storage capacity. Global operations considering the whole model are

composed of local domain operations involving both local data and local processing. Network wide algorithmic rules, which can be applied locally at each stage, control the conversions between local and global data sructures. The latter exists only in a logical sense. Actually its instances are assembled from local elements only when needed, for example for image generation.

The aim of this paper is to introduce the outline of the algorithms needed to establish this kind of a distributed geometric model into a network processor of prefixed structure.

## 2. The centralized approach and network processing

At the base level the geometric entities involved are points, line-segments, surface areas and volumes. These elements must be accessed with attached geometric meanings, and thus special data structures and data base solutions have to be developed for their retrieval and manipulation. Utilizing a general purpose computer as a means of implementation this has been a software design problem in the field of available programming languages and data base construction tools.

The extendible cell idea [1] manages the potentially infinite amount of information by splitting the space into cells recursively until everywhere the information contained in one cell fits into a physical storage element. The physical realization of these storage elements is supposed to be a disk block. The blocks are maintained in a hierarchy by using a spatial directory dividing the study area into a coverage of non-overlapping rectangular cells. The splitting means halving along one coordinate axis only, not by all axes as octtree, where splitting divides the cell automatically into 8 subcells whether needed or not. This introduces a kind of adaptiveness to the data structure as only a necessarily needed amount of cell halvings is done.

This method has a serious drawback for distributed hardware implementation. The cell directory grows in the direction where most entities reside as no balancing mechanism is introduced to the system. In disk realization this means only longer access pointer chains, but introduces no direct physical resource management problems. If we imagine our physical world to consist of a network of separate processors each of which contains its private storage we must simulate the pointer access method by a packet switched network. In processor world the physical connections are rigid and cannot be easily mapped to logical connections (reflecting another architecture) without severely sacrifying the performance. As we do not not know in advance the grow direction of the directory tree, the possibility of getting a severe mismatch between the physical and logical stuctures is obvious.

## 3. The outline of a distributed approach

In order to develop a distributed solution for our problem we must be able to distribute the components of any 3-D model to our architecture. Because of the constraints imposed by the rigid hardware world we cannot adopt the flexible solutions developed for software directly. The idea of using hierarchy for managing the unlimited spatial distribution is a useful concept, but needs modifications for

utilization in fixed environment. The first requirement for the hierarchy system is that it must keep itself in balance in all conditions. The basic idea of B-tree gives the framework for a system which grows in balance. The mechanism for balancing is creation of new intermediate nodes to the tree. In the hardware world this causes similar kinds of allocation problems as mapping of an imbalanced dynamically changing binary tree to our grid network. This is why we have to introduce a new kind of balancing mechanism which avoids the splitting of intermediate nodes. This necessarily then involves transfer of data from one node to the other as a means for balancing instead of node splitting. The processor grid network with reasonably fast interprocessor communication channels can do this in parallel for different parts in the tree supposed the network structure is kept stable.

At the beginning the processor tree is empty and all cell borders are undetermined. As elements (polygons or other simple geometric primitives) are brought into the system, they are first saved at the root node and then gradually transported towards the leaf store cells (processors). As the balance limit of a single processor cell is exceeded, geometric elements from left and right extremes are pushed to the respective son nodes. If the geometric outline of a single element does not fit into the space volume that can be associated with the son cell processor, that piece of geometric information is untransferable and must reside at the parent node whose extents cover the whole volume of interest. In the worst case this node is the root of the tree, which covers the whole space. However, a well behaving geometric model consisting of faceted elements does not contain large amounts of such elements.

Let us inspect the case of breaking the balance limit. Balance limit is a treshold parameter of our processor system. It depends on the state of the system and thus varies by time. It tells the amount of imbalance tolerated by the directory system until balance enforcement procedures are started. Typically this parameter is expressed as ratio between memory load percentages of neighbouring processors. In order to prevent balancing of insignificant loads, a tresholding action is included to the specification of the balance limit. Thus the balance limit index and its associated measures can be expressed as:

```
IF ( imbalance_ratio > balance_limit AND fill_level > treshold )
    THEN take measures for balancing
```

The measures to be taken is transfer of geometric information, i.e. polygons or comparable primitives, between father and son, in both directions depending on imbalance. The element chosen to be transferred is the one nearest to the extent border. The distance measure used in this connection is the minimum offset from the border line of any of the defining points of the element as measured in terms of the dividing coordinate.

The transfer operations lead to the destruction of the accurate border line concept which is valid in space division systems [1]. In most cases the sets of elements are not linearly separable and we cannot find a new border line dividing the space

between these element groups and not splitting any of the elements. As element splitting is prohibited in order to avoid the generation of artificial geometric information [2], we must accept an overlapping directory entry to the father node. It consists of six borders dividing the space with respect to the same coordinate axis: The left and right border line sets (Ll,Lr) and (Rl,Rr) tell the limits of the linearly separable left and right halfspaces (L and R in Figure 1) and the central border set (Cl,Cr) tells the limits of nonseparable common area, later named as the overlapping space (C in Figure 1).

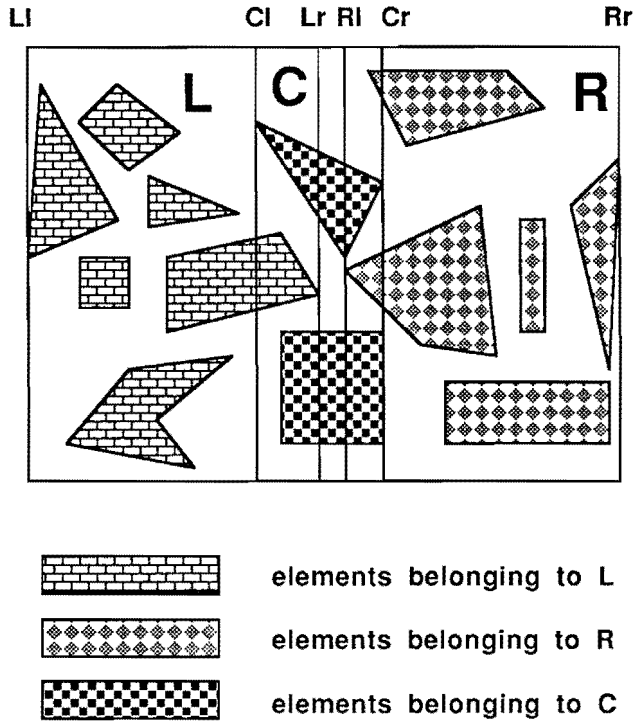


**Figure 1:** The division of geometric elements in the hierarchy with border lines.

By linearly separable elements we mean geometric elements whose extents do not overlap at the coordinate axis of inspection. The meaning of this construct is that geometric information considering the overlapping space between Cl and Cr is logically stored with the directory of the father node, even in case of overflow stored in either of the son nodes. Thus a search procedure for elements at point x is distributed according to the requested coordinates:

```
ASSERT  Ll <= Lr AND  Rl <= Rr AND Cl <= Cr ;
ASSERT  Lr <= Rl AND Cl <= Rl AND Lr <= Cr;
IF Ll < x < Lr THEN search in left subspace;
IF Rl < x < Rr THEN search in right subspace;
IF Cl <  x  < Cr THEN search in overlapping space;
; the third alternative may be valid in addition to
; either of the formers
```

In terms of search operations the handling of elements residing in the overlapping space is costly as it usually needs treatment of two space cells, and must thus be restricted to minimum. An advantage of this schema is that different search routes can be processed in parallel in the tree and the results of the different search paths just have to be combined in the father node.

To formalize the problem for a more detailed analysis it can be simplified to one dimension. Then it reduces to the problem of constructing a B-tree of one-dimensional coordinate intervals (i.e. the extents of geometric elements) as key values. It differs from the construction of a usual ordered binary tree in that intervals are only partially ordered keys because they can overlap.

## 4. The balancing with interval keys

Our problem is to keep in balance a hierarchical network of fixed structure. In this case we do not know anything about the distribution or dynamical behaviour of the incoming data. The fixed structure of the network hierarchy prevents the use of conventional software methods which base on node splitting (B-trees). The space division in our tree structure is determined adaptively by the incoming data elements and the balancing principles mentioned above are obeyed to strive towards a balanced distribution.

In the following simplified analysis we describe each geometric element as a number pair (Xl,Xr) which we associate with the extents of that element in a particular dimension. In a real 3-D situation this means a bounding box of 3 intervals as constraints are given along each coordinate axis. In the hierarchy of processors each node actually considers only one coordinate in space, thus naturally reducing the problem into 1-D. In the 1-D case our sets of border lines are reduced to three number pairs (Ll,Lr), (Rl,Rr) and (Cl,Cr) defining the dimensions of the node and its subtrees. We have already presented a search procedure for elements including a single point, and the search procedure for an interval is an extension of that: instead of searching for any element including a given point we can search for elements in touch with a given interval. In this case the search procedure can be divided into four subcases which we call for search classes in the following:

1) The element is included into the search interval, i.e. the both end points (El,Er) of the searched element E belong to the search interval (Xl,Xr).
2) Only the left endpoint El of the element belongs to interval (Xl,Xr).
3) Only the right endpoint Er of the element belongs to interval (Xl,Xr).
4) The search interval (Xl,Xr) is included to the element (El,Er).

These can be recognized with the following function, which takes two intervals as arguments, the former of which is the search interval X and the latter (A) is the extents of the node possibly containing the element E. The boolean function returns a truth value FALSE if the area cannot contain any acceptable element according to the rules presented for each case.

```
BOOLEAN FUNCTION may_intersect (X,A);
in CASE 1 : RETURN (Al in X OR Ar in X OR Xl in A OR Xr in A)
    ; the intervals A and X intersect
in CASE 2 : RETURN (Xr in A)
    ; right endpoint of search interval X must be in A
    ; because the right end of X must belong to any acceptable E
in CASE 3 : RETURN (Xl in A)
    ; symmetric with case 2
in CASE 4 : RETURN (Xl in A AND Xr in A)
    ; both endpoints of X must belong to any acceptable E
```

Thus for all classes we can write a common procedure using the defined function may_intersect for searching intervals:

```
ASSERT  Ll <= Lr AND  Rl <= Rr AND Cl <= Cr ;
ASSERT  Lr <= Rl AND Cl <= Rl AND Lr <= Cr;
IF may_intersect (X,L) THEN search in left subspace;
IF may_intersect (X,R) THEN search in right subspace;
IF may_intersect (X,C) THEN search in overlapping space;
; if none of the tests is valid the search along that path
; is terminated with the given key pair
```

In order to construct the hierarchy we represent the steps needed for adding a new interval. We begin with an empty tree. No space division is yet determined. As the first element is entered to the tree the border pair (Cl,Cr) of the root is set equal to the first coordinate pair (Nl,Nr) introduced to the system. The subtrees on both sides stay empty and their borders are thus still undetermined. As the balance treshold of the root node is exceeded, the first attempt for balancing is made. In this case it means distribution of the elements gathered in the root further down in the tree. From both sides the elements located at maximum distance from the weight point of the center area are chosen for transportation to the respective son nodes. Both of the sons act as roots of their respective subtrees and handle the initialization and downward balancing measures exactly as their father, the root of the whole tree. The leaf nodes with no sons contain only the center area and cannot balance downwards.

The algorithmic outline is the following:

```
INSERT (Nl, Nr);

ASSERT  Ll <= Lr AND  Rl <= Rr AND Cl <= Cr ;
ASSERT  Lr <= Rl AND Cl <= Rl AND Lr <= Cr;

BEGIN
  put interval in center area;
  IF Cl > Nl THEN Cl := Nl;
  IF Cr < Nr THEN Cr := Nr;
END;

IF imbalance_ratio > balance_limit THEN
BEGIN
  ;choose either the leftmost (Min Xr) or the rightmost (Max Xl)
  ;object of C as the object X

  in CASE (Min Xr) DO
  BEGIN
    IF Xr > Rl THEN RETURN balancing to left not succeeded;
      ;placement to left is impossible due infringement of the
      ;extent of the right subtree
      IF Xr > Lr THEN Lr := Xr;
      ;widen the the right border of the left subtree if needed
      IF Xl < Ll THEN Ll := Xl;
      ;widen the the left border of the left subtree if needed
      INSERT X to the left subtree;
      RETURN successfully balanced to left;
  END;

  in CASE (Max Xl) DO
  BEGIN
    IF Xl < Lr THEN RETURN balancing to right not succeeded;
      ;placement to right is impossible due infringement of the
      ;extent of the left subtree
      IF Xl < Rl THEN Rl := Xl;
      ;widen the the left border of the right subtree if needed
      IF Xr > Rr THEN Rr := Xr;
      ;widen the the right border of the right subtree if needed
      INSERT X to the right subtree;
      RETURN successfully balanced to right;
  END;
END;
```

When there is no room any more for balancing with distribution downwards only, the spatial division of the tree created during the distribution process is adjusted. Geometric elements from the most loaded node are pushed upwards to the center area of the father, which in turn tries to balance by pushing downwards

to the opposite side This procedure involves compressing the borders of the transmitting node, and the element to be pushed upwards is chosen by the criterion of maximum border adjustment.

With all data configurations this may not be possible as suitable elements for downward pushing are not found. In the worst case the center area grows large and for this situation an overflow handling mechanism exists (if we overload the storage capacity of one node).

## 5. The simulation of the network architecture

A simulator is under construction to test this network processor architecture with a real geometric model.

## 6. References

[1] M. Tamminen and R. Sulonen, "The EXCELL Method for Efficient Geometric Access to Data", *19th Design Automation Conference*, Las Vegas, 1982, pp. 345-351.

[2] J. Skytta and T. Takala, "VLSI-based Partially Ordered Tree for Storing 3-D Geometric Information in Data Base", IEEE CompEuro'87: *VLSI and Computers*, Hamburg, Federal Republic of Germany, 11-15 May, 1987, pp 746-749.